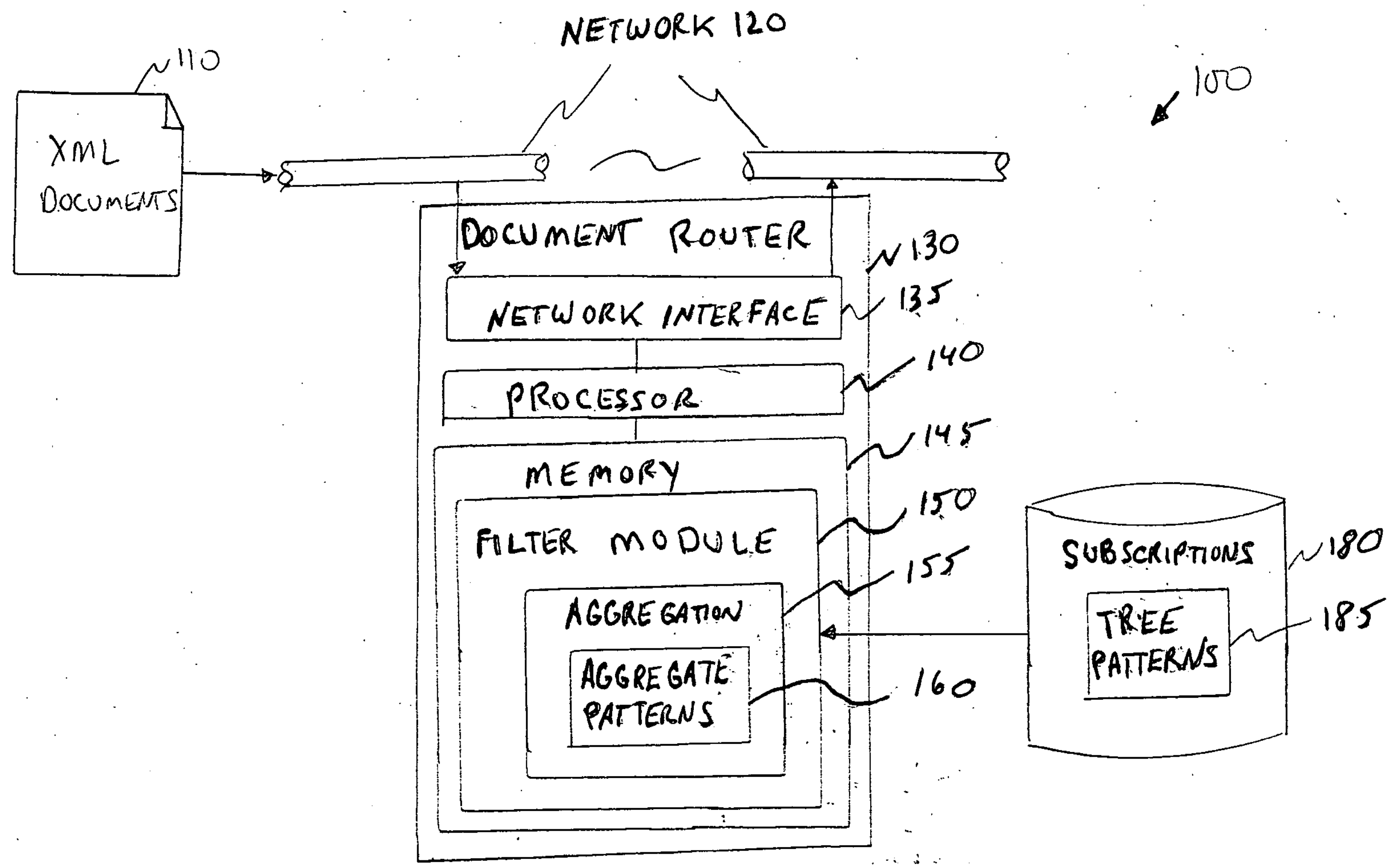
NETWORK 120
100
110
XML DOCUMENTS
DOCUMENT ROUTER
130
NETWORK INTERFACE
135
PROCESSOR
140
MEMORY
145
FILTER MODULE
150
AGGREGATION
155
AGGREGATE PATTERNS
160
SUBSCRIPTIONS
180
TREE PATTERNS
185

FIG. 1

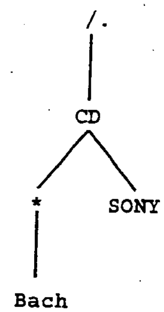

FIG. 2A P_a

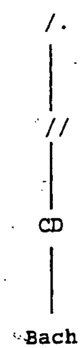

FIG. 2B P_b

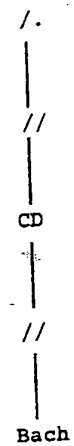

FIG. 2C P_c

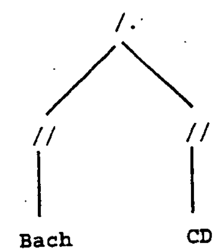

FIG. 2D P_d

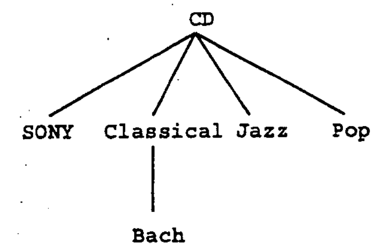

FIG. 2E T

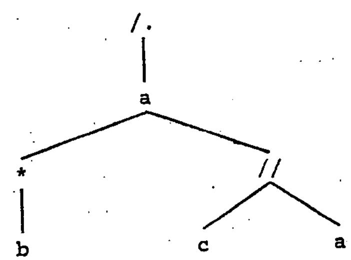

FIG.3A p_a

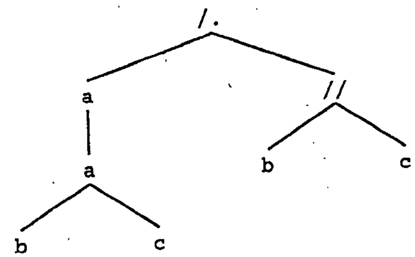

FIG.3B p_b

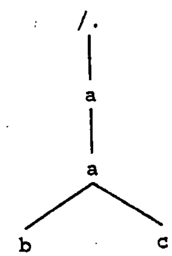

FIG.3C p_c

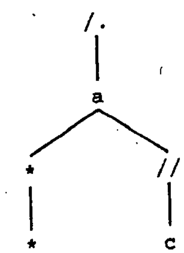

FIG.3D p_d

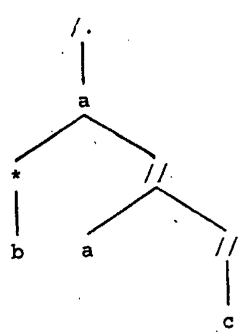

FIG.3E p_e

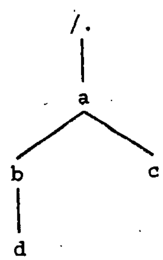

FIG.3F p_f

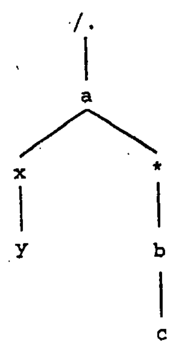

FIG.3G p_g

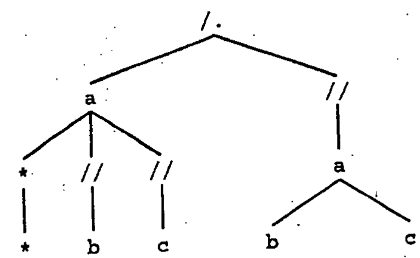

FIG.3H p_h

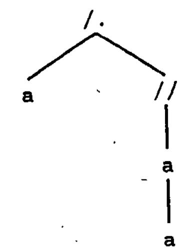

FIG.3I p_i

FIG. 4A

METHOD LUB (p, q)
Input: p and q are tree patterns.
Output: A tree pattern representing the LUB of p and q.
1) **if** $(q \sqsubseteq p)$ **then** return p;
2) **if** $(p \sqsubseteq q)$ **then** return q;
3) Initialize $TCSubPat[v, w] = \emptyset$, $\forall\, v \in Nodes(p),\ \forall\, w \in Nodes(q)$;
4) Let v_{root} and w_{root} denote the root nodes of p and q, resp.;
5) **for** each $v \in Child(v_{root}, p)$ **do**
6)   **for** each $w \in Child(w_{root}, q)$ **do**
7) $TCSubPat[v, w] =$ LUB_SUB $(v, w, TCSubPat)$;
8) Create a tree pattern x with root node label /. and the set of child sub-patterns
$$\bigcup_{v \in Child(v_{root}, p), w \in Child(w_{root}, q)} TCSubPat[v, w];$$
9) **return** MINIMIZE (x);

FIG. 4B

METHOD LUB_SUB $(v, w, TCSubPat)$
Input: v, w are nodes in tree patterns p, q (respectively), $TCSubPat$ is a 2-dimensional array such that $TCSubPat[v, w]$ is the set of tightest container sub-patterns of $Subtree(v, p)$ and $Subtree(w, q)$.
Output: $TCSubPat[v, w]$.
1) **if** $(TCSubPat[v, w] \neq \emptyset)$ **then**
2)   **return** $TCSubPat[v, w]$;
3) **else if** $(Subtree(w, q) \sqsubseteq Subtree(v, p))$ **then**
4)   **return** $\{Subtree(v, p)\}$;
5) **else if** $(Subtree(v, p) \sqsubseteq Subtree(w, q))$ **then**
6)   **return** $\{Subtree(w, q)\}$;
7) **else**
8)   Initialize $R = \emptyset$; $R' = \emptyset$; $R'' = \emptyset$;
9)   **for** each $v' \in Child(v, p)$ **do**
10)   **for** each $w' \in Child(w, q)$ **do**
11) $R = R \cup$ LUB_SUB $(v', w', TCSubPat)$;
12)   **for** each $v' \in Child(v, p)$ **do**
13) $R' = R' \cup$ LUB_SUB $(v', w, TCSubPat)$;
14)   **for** each $w' \in Child(w, q)$ **do**
15) $R'' = R'' \cup$ LUB_SUB $(v, w', TCSubPat)$;
16)   Let x be the pattern with root node label $MaxLabel(v, w)$ and set of child subtree patterns R;
17)   Let x' be the pattern with root node label // and set of child subtree patterns R';
18)   Let x'' be the pattern with root node label // and set of child subtree patterns R'';
19)   **return** $TCSubPat[v, w] = \{x, x', x''\}$;

FIG.5A

METHOD CONTAINS (p, q)
Input: p and q are two tree patterns.
Output: Returns $true$ if $q \sqsubseteq p$; $false$ otherwise.
1) Initialize $Status[v, w] = null$,
   $\forall\, v \in Nodes(p),\ \forall\, w \in Nodes(q)$;
2) Let v_{root} and w_{root} denote the root nodes of p and q, resp.;
3) **if** $(Child(v_{root}, p) = \emptyset)$ **then**
4)   **return** $true$;
5) **else**
6)   **return** CONTAINS_SUB $(v_{root}, w_{root}, Status)$;

FIG.5B

METHOD CONTAINS_SUB $(v, w, Status)$
Input: v, w are nodes in tree patterns p, q (respectively),
   $Status$ is a 2-dimensional array such that each
   $Status[v, w] \in \{null, false, true\}$.
Output: $Status[v, w]$.
1) **if** $(Status[v, w] \neq null)$ **then**
2)   **return** $Status[v, w]$;
3) **if** (v is a leaf node in p) **then**
4) $Status[v, w] = (label(w) \preceq label(v))$;
5) **else if** $(label(w) \npreceq label(v))$ **then**
6) $Status[v, w] = false$;
7) **else**
8) $Status[v, w] =$

$$\bigwedge_{v' \in Child(v,p)} \left(\bigvee_{w' \in Child(w,q)} \text{CONTAINS\_SUB}\ (v', w', Status) \right);$$

9)   **if** $(Status[v, w] = false)$ **and** $(label(v) = //)$ **then**
10) $Status[v, w] =$
   $\bigwedge_{v' \in Child(v,p)}$ CONTAINS_SUB $(v', w, Status)$;
11)   **if** $(Status[v, w] = false)$ **and** $(label(v) = //)$ **then**
12) $Status[v, w] = \bigvee_{w' \in Child(w,q)}$ CONTAINS_SUB $(v, w', Status)$;
13) **return** $Status[v, w]$;

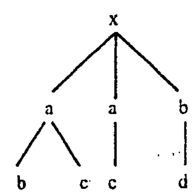

FIG. 6A T1

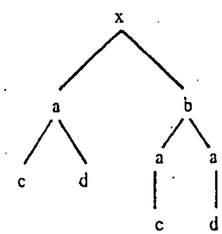

FIG. 6B T2

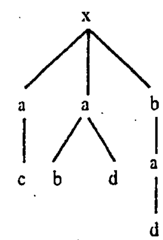

FIG. 6C T3

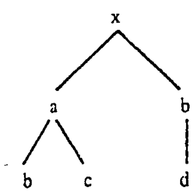

FIG. 6D Skeleton tree for T1

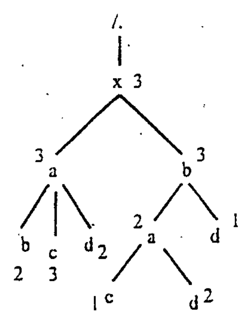

FIG. 6E Document Tree

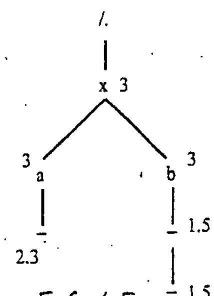

FIG. 6F Compressed Document Tree

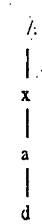

FIG. 6G p1

FIG. 6H p2

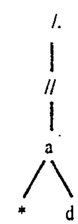

FIG. 6I p3

METHOD SEL(v, t)

Input: v is a node in tree pattern p, t is a node in DT.
Output: $SelSubPat[v, t]$.

if ($SelSubPat[v, t]$ is already computed) **then**
    return $SelSubPat[v, t]$;
else if ($label(t) \not\preceq label(v)$) **then**
    return $SelSubPat[v, t] = 0$;
else if (v is a leaf) **then**
    return $freq(t)/N$;
for each child $v_c \in Child(v, p)$ **do**
    $Sel_{v_c} = \max_{t_c \in Child(t, DT)}\{\mathrm{SEL}(v_c, t_c)\}$;
$Sel = \prod_{v_c \in Child(v,p)} Sel_{v_c}$;
if ($label(v) = //$) **then**
    $Sel_v = \prod_{v_c \in Child(v,p)} \mathrm{SEL}(v_c, t)$;
    $Sel = \max\{Sel, Sel_v\}$;
    $Sel_v = \max_{t_c \in Child(t, DT)}\{\mathrm{SEL}(v, t_c)\}$;
    $Sel = \max\{Sel, Sel_v\}$;
return $SelSubPat[v, t] = Sel$

FIG. 7

METHOD AGGREGATE (S, k)

Input: S is a set of tree patterns, k is a space constraint.

Output: A set of tree patterns S' such that $S \sqsubseteq S'$ and $\sum_{p \in S'} |p| \leq k$.

Initialize $S' = S$;
while $(\sum_{p \in S'} |p| > k)$ **do**
    $C_1 = \{x \mid x = \text{PRUNE}(p, |p| - 1),\ p \in S'\}$;
    $C_2 = \{x \mid x = \text{PRUNE}(p \sqcup q, |p| + |q| - 1),\ p, q \in S'\}$;
    $C = C_1 \cup C_2$;
    Select $x \in C$ such that $Benefit(x)$ is maximum;
    $S' = S' - \{p \mid p \sqsubseteq x, p \in S'\} \cup \{x\}$;
return S';

FIG. 8